Energy, Mines and Resources Canada
Énergie, Mines et Ressources Canada

CANMET
Canada Centre for Mineral and Energy Technology
Centre canadien de la technologie des minéraux et de l'énergie

MAP DISPLAY PROGRAM SOFTWARE

V.Srajer, N.J.Stuart, M.Heximer, W.Veldhuis, C.Neumiller

March 1987

(CANMET)

(CANMET)

20 pp

MICROMEDIA

ENERGY RESEARCH PROGRAM
COAL RESEARCH LABORATORIES
DIVISION REPORT ERP/CRL 87-17 (TR)

87-05147/

MAP DISPLAY PROGRAM SOFTWARE

by

V. Srajer*, N.J. Stuart**, M. Heximer***, W. Veldhuis***, C. Neumiller***.

ABSTRACT

In this report, two sets of programs, that were developed as part of the overall deterministic truck and shovel model, are described. The first program, "RMAP" will display mine haulage networks created by digitizing at the VT 240 terminal. The second program called "VMAP" will display similar network at the Vaxstation 100 graphics screen.

The step-by-step instruction and execution of the program is explained. A listing of both programs in Fortran 77 is enclosed for further reference.

---

* Head, Operations Research, CANMET, CRL, Surface Mining Lab., Devon, Alb.
** Research Scientist, CANMET, CRL, Surface Mining Lab., Devon, Alberta.
*** Co-Op Students, University of Alberta, Edmonton, Alberta.

PROGRAMME D'AFFICHAGE MAP

par

V. Srajer*, N.J. Stuart**, M. Heximer***, W. Veldhuis***, C. Neumiller***.

RÉSUMÉ

Dans le présent rapport, on décrit deux ensembles de programmes qui ont été écrits dans le cadre du modèle déterministe global du camion et de la pelle. Le premier programme, "RMAP", affiche des réseaux de transport dans les mines créés par numérisation au terminal VT 240. Le deuxième programme, "VMAP", affiche des réseaux similaires sur l'écran graphique Vaxstation 100.

On explique les instructions et l'exécution pas à pas du programme. Une liste de deux programmes écrits en Fortran 77 est incluse pour fins de référence.

*Chef, Recherche opérationnelle, CANMET, LRC, Lab. d'exploitation en surface, Devon (Alb.).
**Chercheur scientifique, CANMET, LRC, Lab. d'exploitation en surface, Devon (Alberta).
***Étudiants inscrits à un programme de coopération, Université de l'Alberta, Edmonton (Alberta).

CONTENT

	Page
ABSTRACT	i
RESUME	iii
INTRODUCTION	1
RMAP PROGRAM	1
How to Use Program	1
Available Options	1
Deletions of Files	2
Program Listing	3
VMAP PROGRAM	12
How to Use Program	12
Conversion of Files	12
Program Listing	12
CONCLUSION	16
ACKNOWLEDGEMENT	16
REFERENCES	16

INTRODUCTION

Two programs were written in VAX Fortran 77. One program will display mine layouts created for a VT-240 terminal, the other will display it on a Vaxstation 100. The software created for execution of both programs is explained and listed in the report.

RMAP PROGRAM

RMAP is a program written in VAX Fortran 77 utilizing REGIS for the VT-240 terminal. The program will explain mine layouts that have been created by DIGIT (1) with the capability of creating and displaying haulage routes. The DIGIT output file must be converted to another form to be used by RMAP, this is done by the program DAT_REGIS.

Once a mine layout has been digitized with DIGIT (1), DAT_REGIS is run. This program will prompt you for the name of the input file, so enter the name of the DIGIT output file. DAT_REGIS will reformat the data into a form useable by RMAP. It will also write the filename into a directory used by RMAP called MINES.RMAP. The RMAP is now ready to be executed on any VT-240 terminal.

How to Use Program

The program is currently executed by the statement:

```
RUN [IAN.FALL86.REGIS]RMAP
```

Once RMAP has been executed, a list of available mine sites will be displayed. Enter the number corresponding to the mine that you wish to work with.

The mine layout will be displayed on the left side of the screen and the menu will be displayed on the right side of the screen. The menu options are:

1. SELECT NEW MINE
2. DISPLAY SEGMENTS / CREATE ROUTES
3. DISPLAY ROUTES
4. EXIT PROGRAM

Type in the number of the option you wish to use.

Available Options.

The options identified are now explained.

Option 1 : **SELECT NEW MINE**

This option will display the list of available mine sites and ask for the number of the mine that is to be viewed. This option allows work with different maps without exiting the program.

At the present time, mine haulage routes (status Jan. 1, 1986) are available from the following mines: Quintette, Gregg River, Coal Valley, Luscar, Highvale, Whitewood, Line Creek, Coal Mountain, Harmer.

Option 2 : **DISPLAY SEGMENTS / CREATE ROUTES**

This option will display all of the nodes in the mine site. The user is asked to select a starting node. To start, enter one of the nodes displayed or press <CR> to exit this option. If the node that was entered was not found on the map, the user is asked to enter a valid node. The node names must be entered exactly as they are displayed on the screen. A list of the connecting nodes will be displayed on the screen, enter one of these or <CR> to end route. The path list of the route will be displayed on the right side of the screen and the segments will be highlighted on the map in red. If the user wishes to delete the last segment from the route, the previous node name from the route list must be entered. The route list is located on the right hand side of the screen.

After the route is terminated with <CR>, the program will prompt the user with the option to save the route. If "Y" is entered, the User will be asked to enter the name of the route. It is recommended to use a name that is a combination of the starting and ending nodes of the route. This route will then be appended to the end of the mine site file. If "N" is entered, the route will not be saved for later use. The user is then asked if he wishes to return to the main menu. If "N" is entered, the program will prompt you to enter a new route. The maximum number of saved routes is 40 and the maximum number of nodes in a route is also 40.

Option 3 : **DISPLAY ROUTES**

This option will display the saved routes from Option 2, if there are any. The route that should be displayed may be now entered. The route path will then be displayed on the right hand side of the screen under the route name, and the route will be highlighted in red. The user is then asked if he wishes to return to main menu. If "N" is entered, you are asked to enter another route.

Option 4 : **EXIT PROGRAM**

This option is used to exit the program. It will close all files used by the program, it will also reset the screen and exit REGIS mode.

<u>Deletion of Files.</u>

The user may also choose to delete a mine file from the RMAP directory. To delete a file, a MINES.RMAP file must be edited. This is done by deleting the line which contains the undesired mine name. The mine file will then be

deleted from the directory. The whole file called FILENAME.REG can be deleted if there is no need to save it for the future use.

If a route is deleted that was saved previously, a FILENAME.REG is edited. You must go to the end of the file and delete the line containing the undesired route.

Program listing.

The program listing using Vax Fortran 77 is as follows:

```
      PROGRAM RMAP
c     LINK RMAP+RGRAPHICS

c     Program will display mine layout created by DIGIT program
c       on VT240 terminal, allow the highlighting and building of
c       haulage routes. Output from DIGIT must be processed through
c       DAT_REGIS program.

      integer i,nnode,nseg,npts(60),ntitle,nunits,nrnodes(40)
      integer nroutes
      real x(60,100),y(60,100),xfirst,xmax,yfirst,ymax
      real xnorth,ynorth,fact,range,inc,ang
      character*8 head(60),tail(60),noden(60),route(40),rnodes(40,40)
      character units*20,title*60,esc*1,but*1

c     i       = loop counter
c     nnode   = number of nodes in mine layout
c     nseg    = number of segments in mine layout
c     npts    = array containing the number points in each segment
c     ntitle  = number of characters in title
c     nunits  = number of characters in units
c     nrnodes = array containing the number of nodes in each saved route
c     nroutes = number of saved routes
c     x,y     = 2-D array containing the coordinates of the segments
c     xfirst,
c     yfirst  = starting values of x & y axis
c     xmax,
c     ymax    = maximum values of x & y axis
c     range   = range of values that axis cover
c     inc     = distance between axis tick marks in map units
c     fact    = pixel conversion factor to convert from true values
c               to screen coordinates
c     head,
c     tail    = character arrays containing the names of starting and ending
c               nodes for segments
c     noden   = node names used for display and checking purposes
c     route   = saved route names
c     rnodes  = 2-D array that contains the nodes that make up
c               each saved route
c     units   = character variable containing units for axis
c     title   = character variable containing the mine title
c     esc     = contains the escape code 27 used for regis calls
c     but     = used for determination of when lines end

c     Initialize screen and scroll protect.
```

```
      esc=char(27)
      write(*,*) esc,'[23;24r'
      write(*,*) esc,'P1p'
      write(*,*) 's(a[0,479][799,0])(e)'
      write(*,*) 's(m1(ar)2(ah20165s60)3(ah240150s60))'
      write(*,*) esc,'\'
      do 5 i=1,20
            write(*,*) esc,'[B'
5     continue

c     Open selected map file for input

7     call assign_file

c     Read in data from file

      read(1,'(1x,a60)') title
      call charcount(title,60,ntitle)
      read(1,*) xfirst,xmax
      read(1,*) yfirst,ymax

      read(1,*) xnorth,ynorth

      read(1,'(1x,a20)') units
      call charcount(units,20,nunits)

      nseg=1
10    read(1,'(1x,a8,1x,a8)') head(nseg),tail(nseg)
      n=1
20    read(1,'(1x,a1,2f9.2)') but,x(nseg,n),y(nseg,n)

c     Conversion from real to screen co-ordinates

      x(nseg,n)=(x(nseg,n)-xfirst)*375./(xmax-xfirst)+50.
      y(nseg,n)=(y(nseg,n)-yfirst)*375./(ymax-yfirst)+79.

c     Check for end of line segment

      if(but.ne.'D'.and.but.ne.'#') then
            n=n+1
            goto 20
      endif
      npts(nseg)=n
      if(but.ne.'#') then
            nseg=nseg+1
            goto 10
      endif

c     Read in node names

      read(1,*) nnode
      do 30 i=1,nnode
            read(1,'(1x,a8)') noden(i)
30    continue

c     Read in routes

      do 40 i=1,40
35      format(1x,a8,i3,40a8)
        read(1,35,end=50) route(i),nrnodes(i),(rnodes(i,k),k=1,nrnodes(i))
40    continue

50    nroutes=i-1
```

```
c       Calculate parameters for axis and screen

        range=xmax-xfirst
        fact=range/375.
        inc=250.
        if(range.gt.2500.) inc=1000.
        if(range.gt.10000.) inc=2500.

c       Enter Regis and black out screen

        write(*,*) esc,'P1p'
        write(*,*) 'w(i(d))p[0,0]f(v(b)[+799][,+479][-799](e))'
        write(*,*) 'w(i(r))p[50,454]v[+375][,-375]'

c       Call subroutines to create axis

        call axis(50.,79.,-375.,0.,units,xfirst,fact,inc)
        call axis(50.,79.,375.,90.,units,yfirst,fact,inc)

        write(*,*) 't(s1)[10,0]'
        write(*,*) 'w(i(g))p[',212-ntitle*5,',474]t"',title(1:ntitle),'"'

c       Draw out line segments from array

        do 60 i=1,nseg
                call draw_seg(x,y,i,npts)
60      continue

c       Create menu for program branching

100     call draw_menu(i)
        if(i.eq.1) then
                close(1)
                write(*,*) 'w(i(d))p[0,0]f(v(b)[+799][,+479][-799](e))'
                write(*,*) esc,'\'
                goto 7
        endif
        if(i.eq.2) call dis_node(noden,nnode,x,y,npts,head,tail,nseg,
        3                        nroutes,nrnodes,route,rnodes)
        if(i.eq.3) call dis_routes(x,y,npts,nseg,head,tail,route,nroutes,
        3                          nrnodes,rnodes)
        if(i.eq.4) then
                write(*,*) 's(e)',esc,'\',esc,'[1;24r'
                close(1)
                stop
        endif
        goto 100

        end

**********************************

        subroutine draw_seg(x,y,nseg,npts)

c       Subroutine to draw segments requires segment number in array

        integer nseg,npts(60)
        real x(60,100),y(60,100)

        write(*,*) 'p[',x(nseg,1),',',y(nseg,1),']'

c       Check for straight line or curve
```

```
        if(npts(nseg).eq.2) then
                write(*,*) 'v[',x(nseg,2),',',y(nseg,2),']'
        else
                write(*,*) 'c(s)[]'
                do 100 i=1,npts(nseg)
                        write(*,*) '[',x(nseg,i),',',y(nseg,i),']'
100             continue
                write(*,*) '[](e)'
        endif

        return
        end

*************************************

        subroutine draw_menu(i)

c       Subroutine to draw menu and accept menu choice

        integer i
        character*1 esc

        esc=char(27)

        call black_box
        write(*,*) 'p[600,400]t"MENU"p[600,381]v[+40]'
        write(*,*) 'p[500,320]t"<1> SELECT NEW MINE"'
        write(*,*) 'p[500,280]t"<2> DISPLAY SEGMENTS"'
        write(*,*) 'p[500,250]t"   /CREATE ROUTES"'
        write(*,*) 'p[500,210]t"<3> DISPLAY ROUTES"'
        write(*,*) 'p[500,170]t"<4> EXIT PROGRAM"'
        write(*,*) esc,'\'
200     write(*,*) ' Enter menu choice :'
        read(*,*) i
        if(i.gt.4.or.i.lt.1) goto 200
        write(*,*) esc,'P1p'
        call black_box

        return
        end

*******************************

        subroutine black_box

c       Subroutine to fill in area to left of map

        write(*,*) 'w(i(d))p[440,40]'
        write(*,*) 'f(v(b)[+359][,+439][-359](e))w(i(g))'

        return
        end

*******************************

        subroutine dis_node(noden,nnode,x,y,npts,head,tail,nseg,
        3                   nroutes,nrnodes,route,rnodes)

c       Subroutine to display nodes and build routes

        integer nnode,xpos,ypos,icount,ip,i,nseg,npts(nseg),enlen(10)
        integer nroutes,nrnodes(40)
        real x(nseg,100),y(nseg,100)
```

```
      character*8 noden(nnode),esc*1,head(nseg),tail(nseg)
      character*8 enode(10),pnode(60),route(40),rnodes(40,40)

c     enlen   = array containing no. of characters in enode
c     enode   = array of node names denoting connecting nodes
c     icount  = no. of connecting nodes
c     ip      = no. of nodes in path
c     pnode   = array of node names in path

      esc=char(27)
      write(*,*) 'w(i(g))p[520,478]t"MINE NODES"p[520,459]v[+100]'

c     Write out node names to screen

      xpos=445
      ypos=455
      do 100 i=1,nnode
            write(*,*) 'p[',xpos,',',ypos,']t"',noden(i),'"'
            ypos=ypos-20.
            if(ypos.lt.60) then
                  ypos=455
                  xpos=xpos+90
            endif
100   continue
      write(*,*) 'w(i(r))p[710,478]t"ROUTE"p[710,459]v[+50]'

      write(*,*) esc,'\'

c     Prompt for creation of new route

150   write(*,*) 'Enter starting node  (<cr> to exit):'
      ip=1
      read(*,'(a8)') pnode(ip)
      if(pnode(ip)(1:3).eq.'   ') goto 700
c     Check for valid node name
      do 200 i=1,nnode
            if(pnode(1).eq.noden(i)) goto 225
200   continue
      write(*,*) ' Node not found, try again.(<cr> to continue)'
      accept *
      goto 150

225   write(*,*) esc,'P1p'

c     Erase route list and write out new route list

250   write(*,*) 'w(i(d))p[715,455]f(v(b)[+84][,-415],[-84](e))w(i(r))'
      ypos=455
      do 275 i=1,ip
            write(*,*) 'p[715,',ypos,']t"',pnode(i),'"'
            ypos=ypos-20
275   continue
      write(*,*) esc,'\'
      icount=0

c     Search for connecting nodes

      do 300 i=1,nseg
            if(pnode(ip).eq.tail(i)) then
                  icount=icount+1
                  enode(icount)=head(i)
            endif
            if(pnode(ip).eq.head(i)) then
```

```
                        icount=icount+1
                        enode(icount)=tail(i)
                endif
300     continue

c       Determine length of node names and display connecting nodes

        do 305 i=1,icount
                call charcount(enode(i),8,enlen(i))
305     continue
310     write(*,311) (enode(k)(1:enlen(k)),k=1,icount)
311     format(' Select next node (<cr> to exit):',10(1x,a))
        ip=ip+1
        read(*,'(a8)') pnode(ip)
        if(pnode(ip)(1:3).eq.'   ') goto 500

c       Check for valid connecting node name

        do 400 i=1,icount
                if(pnode(ip).eq.enode(i)) goto 425
400     continue
        write(*,*) ' Node ',pnode(ip),' not attached to node ',pnode(ip-1)
        accept *
        ip=ip-1
        goto 310
425     write(*,*) esc,'P1p'

c       Highlight new segment in route if not rerouting
        if(ip.gt.2.and.pnode(ip).eq.pnode(ip-2)) then
                write(*,*) 'w(i(G))'
                call check_seg(pnode(ip-1),pnode(ip),x,y,npts,head,tail,nseg)
                ip=ip-2
        else
                write(*,*) 'w(i(b))'
                call check_seg(pnode(ip-1),pnode(ip),x,y,npts,head,tail,nseg)
        endif
        goto 250

c       Prompt for route saving

500     ip=ip-1
        write(*,*) ' Do you wish to save this route (Y/N) ?'
        read(*,'(a1)') prompt
        if(prompt.eq.'y'.or.prompt.eq.'Y') then
                write(*,*) ' Enter route name '
                read(*,'(a8)') route(nroutes+1)
                nroutes=nroutes+1
                write(1,580) route(nroutes),ip,(pnode(k),k=1,ip)
580             format(1x,a8,i3,40a8)
                nrnodes(nroutes)=ip
                do 585 i=1,ip
                        rnodes(nroutes,i)=pnode(i)
585             continue

        endif
        write(*,*) esc,'P1p'
        write(*,*) 'w(i(d))p[715,455]f(v(b)[+84][,-415],[-84](e))w(i(g))'
c       Redraw over route in original colour
        do 600 i=1,ip-1
                call check_seg(pnode(i),pnode(i+1),x,y,npts,head,tail,nseg)
600     continue
        write(*,*) esc,'\'
700     write(*,*) ' Return to main menu (Y/N) ?'
```

```
        read(*,'(a1)') prompt
        if(prompt.ne.'Y'.and.prompt.ne.'y') goto 150
        write(*,*) esc,'P1p'
        return
        end

*************************************************

        subroutine check_seg(snode,enode,x,y,npts,head,tail,nseg)

c       Subroutine used to determine if given nodes are
c       a segment and will draw the segment

        integer nseg,npts(nseg)
        real x(nseg,100),y(nseg,100)
        character*8 snode,enode,head(nseg),tail(nseg)

c       snode   = starting node
c       enode   = ending node

        do 100 i=1,nseg
                if(snode.eq.head(i).and.enode.eq.tail(i)) goto 200
                if(enode.eq.head(i).and.snode.eq.tail(i)) goto 200
100     continue
        return
200     call draw_seg(x,y,i,npts)
        return
        end

*************************************************
        subroutine dis_routes(x,y,npts,nseg,head,tail,route,nroutes,
        3                     nrnodes,rnodes)

c       Subroutine to display saved routes

        integer nseg,npts(nseg),nroutes,nrnodes(40),xpos,ypos,i,k,nchar
        real x(nseg,100),y(nseg,100)
        character*8 head(nseg),tail(nseg),route(40),rnodes(40,40)
        character rname*8,esc*1,prompt*1

c       rname   = route name from keyboard
c       prompt  = prompt variable

        esc=char(27)

c       Display route names

        write(*,*) 'w(i(g))p[520,478]t"ROUTES"p[520,459]v[+60]'

        if(nroutes.eq.0) then
                write(*,*) 'p[445,455]t"NO ROUTES AVALIABLE"'
                write(*,*) esc,'\'
                write(*,*) ' Press <cr> to return to main menu'
                accept *
                write(*,*) esc,'P1p'
                return
        endif

        ypos=455
        xpos=445

        do 100 i=1,nroutes
                write(*,*) 'p[',xpos,',',ypos,']t"',route(i),'"'
```

```
                ypos=ypos-20
                if(ypos.lt.60) then
                        ypos=455
                        xpos=xpos+90
                endif
100     continue

130     write(*,*) esc,'\'

c       Input route name and check validity

150     write(*,*) ' Enter route name to be displayed'
        read(*,'(a8)') rname

        do 200 i=1,nroutes
                if(rname.eq.route(i)) goto 300
200     continue
        write(*,*) ' Route not found, Try again (<cr> to continue)'
        accept *
        goto 150

c       Display route path and highlight map

300     write(*,*) esc,'P1p'
        write(*,*) 'v(i(r))p[715,478]t"',route(i),'"'
        call charcount(route(i),8,nchar)
        write(*,*) 'p[715,459]v[+',nchar*10,']'
        ypos=455
        do 400 k=1,nrnodes(i)
                write(*,*) 'p[715,',ypos,']t"',rnodes(i,k),'"'
                ypos=ypos-20
100     continue
        write(*,*) 'w(i(h20145s100))'

        do 450 k=2,nrnodes(i)
         call check_seg(rnodes(i,k-1),rnodes(i,k),x,y,npts,head,tail,nseg)
450     continue
        write(*,*) esc,'\'
        write(*,*) ' Return to main menu (Y/N) ?'
        read(*,'(a1)') prompt
        write(*,*) esc,'P1p'
        write(*,*) 'w(i(g))'
        do 500 k=2,nrnodes(i)
          call check_seg(rnodes(i,k-1),rnodes(i,k),x,y,npts,head,tail,nseg)
500     continue
        if(prompt.eq.'y'.or.prompt.eq.'Y') return
        write(*,*) 'w(i(d))p[714,479]f(v(b)[+75][,-439][-75](e))'
        goto 130
        end

************************************

        subroutine assign_file

c       Subroutine to open file

        integer i,nchar,xpos,ypos,nnames
        character*20 mapnames(66),fname,esc*1,num*2

c       mapnames  = array containing map file names
c       fname     = selected map file

        esc=char(27)
```

```
      open(unit=3,file='mines.rmap',status='old')

c     Read in map file names

      do 100 i=1,66
            read(3,'(1x,a20)',end=200) mapnames(i)
100   continue

200   close(3)

c     Display numbered map files available

      nnames=i-1
      write(*,*) esc,'P1p'
      write(*,*) 'w(i(g))p[350,478]t"MINES"p[350,459]v[+50]'
      write(*,*) 't(s1)[10,0]'
      xpos=5
      ypos=455
250   do 300 i=1,nnames
            write(num,'(i2)') i
            write(*,*) 'w(i(g))p[',xpos,',',ypos,']t"[',num,']"'
            write(*,*) 'w(i(r))p[',xpos+50,',',ypos,']t"',mapnames(i),'"'
            ypos=ypos-20
            if(ypos.lt.40) then
                  xpos=xpos+260
                  ypos=455
            endif
300   continue
      write(*,*) esc,'\'

c     Input selected file name

350   write(*,*) ' Enter file number'
      read(*,'(a20)') fname

      if(ichar(fname(1:1)).lt.49.or.ichar(fname(1:1)).gt.57) then

            write(*,*) ' Not a number'
            goto 350
      endif

      read(fname,*) i
      call charcount(mapnames(i),20,nchar)
      open(unit=1,file=mapnames(i)(1:nchar)//'.reg',status='old')
      return
      end
```

VMAP PROGRAM

The program VMAP display mine sites that have been digitized by the DIGIT program on the Vaxstation 100. The output file from DIGIT must be converted into a different format to be useable by VMAP. This is done by the program DAT_VSTA. The high resolution graphics capability of the Vaxstation combined with a "Zoom" capability allows for detailed viewing of mine haulage systems. The high resolution also results in the printing of node names and text on the screen in better form then on VT-240 terminal.

How to Use Program.

The program is executed by typing:

RUN [IAN.FALL86.VSTA]VMAP

The user is prompted for an input file name. "DIR.VST" should then be typed to check which files are valid input for the VMAP program. Type in the file name excluding the .VST suffix.

The map will then be displayed on the screen along with the axes and north arrow. The mouse may be used to move the window to the top right hand corner of the screen. Answer "Y" or "N" in response to the prompt of "zoom". If "Y" is entered, enter the x and y coordinates of the lower left hand corner of the proposed "zoom" area, then enter the x and y coordinates of the upper right hand corner of the "zoom" area.

After the viewing of the zoomed area is completed, the mouse may be used again to move the window out of view.

The User is then asked if he wish to "zoom" again, until the response is "N". The program will then be terminated after <CR> is pressed.

Conversion of Files.

The DAT files produced by DIGIT may be converted to VMAP input files by the command:

Run DAT_VSTA.

The program DAT_VSTA is executed. The DIGIT output file name is then entered, excluding the .DAT suffix. The program will create a VMAP compatible file with .VST suffix.

Program Listing.

The program listing using Vax Fortran 77 is as follows:

```
      PROGRAM VMAP

c     LINK VMAP+VGRAPHICS
c     The program is designed to display converted DIGIT output
c     files onto a vaxstation 100.  It creates a mine layout
c     with axes, node names, additional text, and north arrow

      real x(1000),y(1000),xn(200),yn(200)
      real xmin,ymin,xmax,ymax,dx,dy,range,fact
      integer n,k,j,i,lin,ncount,chan1,chan2,pb1,pb2,win1,win2
      character*20 wname1,wname2,xaxis,yaxis,but,linename
      character fname*30
      character*20 noden(200)
      logical flag

      write(*,*) ' enter the name of the input file'
      read(*,'(a)') fname
      call charcount(fname,lin)

      open(unit=1,file=fname(1:lin)//'.vst',status='old')

      call cgl$initialize_core()
      read(1,20) wname1
20    format(1x,a60)
c     read in x maximum and minimum
      read(1,*) xfirst,xmax
c     read in y maximum and minimum
      read(1,*) yfirst,ymax
      range=xmax-xfirst
      fact=range/750.
      ymin=yfirst-fact*50.
      xmin=xfirst-fact*50.
      xmax=xmax+fact*25.
      range=xmax-xmin
      fact=range/800.
      flag=.true.
      call dmap(chan1,pb1,win1,wname1,xmin,ymin,xfirst,yfirst,range,fact,
     3          flag)
888   close(unit=1)
      write(*,*) ' Do you want to zoom (Y/N)'
      read(*,'(a1)') prompt
      if(prompt.ne.'Y'.and.prompt.ne.'y') goto 999
      open(unit=1,file=fname(1:lin)//'.vst',status='old')
      read(1,20) wname2
      read(1,*) xmin,xmax
      read(1,*) ymin,ymax
      write(*,*) 'Enter lower left coordinates (x,y) of zoom'
      read(*,*) xmin,ymin
      write(*,*) 'Enter upper right coordinates (x,y) of zoom'
      read(*,*) xmax,ymax
      if(xmax-xmin.gt.ymax-ymin) then
        range=xmax-xmin
      else
        range=ymax-ymin
      endif
      fact=range/800.
      flag=.false.
      call dmap(chan2,pb2,win2,wname2,xmin,ymin,xfirst,yfirst,range,fact,
     3          flag)
      goto 888
999   write(*,*) ' Press <RETURN> to exit'
      accept*
```

```
        close(unit=1)
        call cgl$terminate_core()
        call exit
        stop

        end
*****************************************

        subroutine dmap(chan,pb,win,wname,xmin,ymin,xfirst,yfirst,range,fact,
        3               flag)

c       Subroutine that draws mine segments, node names, and axes.

        real x(1000),y(1000),xn(200),yn(200),xtxt(200),ytxt(200)
        real xmin,ymin,xmax,ymax,dx,dy,range,fact
        real xfirst,yfirst,xlast,ylast,inc
        integer n,k,j,i,lin,ncount,chan,pb,win,ntext,nnodes
        character*20 wname,xaxis,yaxis,but,linename
        character*20 noden(200)
        logical flag
        call objs(800,800,chan,pb,win,wname)
        stat=vsta$select_display(chan)
        call cgl$set_origin(0)
        call cgl$set_window(xmin,xmin+range,ymin,ymin+range)
c       read in data direction
        read(1,*) dx,dy
30      format(1x,a)
c       draw north arrow for big map
        if(fact.ne.0.0) then
          xlast=xfirst+725.*fact
          ylast=yfirst+725.*fact
          dis=sqrt(dx**2+dy**2)
          x1=xlast-100.*fact
          y1=ylast-100.*fact
          x2=x1+(dx/dis)*50.*fact
          y2=y1+(dy/dis)*50.*fact
          call cgl$set_charsize(8.,8.)
          call cgl$set_charspace(8.*fact,0.)
          call arrow(x1,y1,x2,y2)
          call cgl$move_abs_2(x2,y2+20.*fact)
          call cgl$text(%ref('N'),1)
        endif
c       read in plot units and assign to x and y axis' names
c       send to output file as well

        read(1,50) xaxis
50      format(1x,a60)
        yaxis=xaxis
        if(fact.ne.0.) then
          inc=250.
          if(range.gt.2500.) inc=inc*2.
          if(range.gt.5000.) inc=inc*2.
          call axis(xfirst,yfirst,-725.,0.,xaxis,fact,inc)
          call axis(xfirst,yfirst,725.,90.,yaxis,fact,inc)
          call axis2(xfirst,ylast,-725.,0.,fact,inc)
          call axis2(xlast,yfirst,725.,90.,fact,inc)
        endif


c       read in line name, send to output file, and assign to graph title

73      read(1,30) linename
c       initialize n
```

```
        n=1
c       read in point data
100     read(1,10,end=700),but,x(n),y(n)
10      format(1x,a1,2f9.2)
c       check to see if end of line
        if((but.eq.'D').or.(but.eq.'$')) goto 700
        n=n+1
        goto 100

700     continue
        if(n.gt.2) then
          call cgl$curve_abs_2(x,y,n,0)
        else
          call cgl$move_abs_2(x(1),y(1))
          call cgl$polyline_abs_2(x,y,n)
        endif
        if(but.eq.'D') goto 73
        read(1,*) nnodes
        do 800 i=1,nnodes
          read(1,*) xn(i),yn(i)
800     continue
        call cgl$set_marker_symbol(3, ,)
        call cgl$polymarker_abs_2(xn(1),yn(1),nnodes)
        do 950 i=1,200
900       read(1,905,end=999) noden(i),xtxt(i),ytxt(i)
905       format(1x,a20,2f9.2)
          call charcount(noden(i),lin)
          call cgl$set_charsize(8.,8.)
          call cgl$set_charspace(8.*fact,0.)
          call cgl$move_abs_2(xtxt(i),ytxt(i))
          call cgl$text(%ref(noden(i)),lin)
950     continue
999     ntext=i-1
        return
        end
*********************************

        subroutine charcount(char,n)
        character*(*) char
        integer n,i
        do 100 i=1,20
          if(char(i:i).ne.' ') n=i
100     continue
        return
        end

*****************************************

        subroutine objs(w,h,chan,pb,win,wname)
        implicit integer (a-w)
        character wname*(*)

        stat= vsta$create_display(w,h,1,0,16,0,0,chan)
        stat= vsta$create_pasteboard(w,h,pb)
        stat= vsta$paste_display(chan,pb,0,0)
        stat=vsta$create_window(w,h,pb,0,0,win,wname)
        return
        end
```

CONCLUSION

The RMAP and VMAP programs were developed to assist user's of the deterministic truck/shovel model, by providing them with terminal graphics capability, to display haulage routes and control movement of the trucks along these routes. The programs will also provide a visual check of alternative routes, displayed on the screen.

The programs presented in this report are two of the first set of programs developed by the Operations Research group, to provide optimization tools to the surface coal mines that supplied background information for haulage road display and cooperated in defining their requirements for usage of the program.

ACKNOWLEDGEMENT

The cooperation of Quintette, Gregg River, Coal Valley, Luscar, Highvale, Whitewood, Line Creek, Coal Mountain, Harmer mines that provided their mine plans for digitizing and supplied information on their requirements is greatly acknowledged.

REFERENCES

1. C. Neumiller, N.J. Stuart, M. Heximer, W. Veldhuis : "Digitizing Programs to Generate Input Data for a Haulage Simulation Model by Using Fitting Cubic Curves", ERP/CRL Divissional Report 86-129 (TR), CANMET, CRL, Devon, Alberta.